Python: module cdms.cache

cdms.cache

index

CDMS cache management and file movement objects

Modules

cdms.cdmsobj	os	tempfile
cdms.cdurllib	shelve	time
errno	sys	urlparse

Classes

Cache

class ***Cache***

```
# A simple data cache
```

Methods defined here:

__init__(self)

clean(self)

```
Clean pending read notifications.
```

copyFile(self, fromURL, filekey, lcpath=None, userid=None, useReplica=None)

```
Copy the file <fromURL> into the cache. Return the result pat

For request manager transfers, lcpath is the logical collecti
<userid> is the string user ID, <useReplica> is true iff the
search the replica catalog for the actual file to transfer.
```

delete(self)

```
Delete the cache.
```

deleteEntry(self, filekey)

```
Delete a cache index entry.
```

get(self, filekey)

```
Get the path associated with <filekey>, or None if not presen
```

getFile(self, fromURL, filekey, naptime=5, maxtries=60, lcpath=None, userid=None, useReplica=N

Get the file with <fileURL>. If the file is in the cache, rea
If another process is transferring it into the cache, wait fo
transfer to complete. <naptime> is the number of seconds betw
retries, <maxtries> is the maximum number of retries.
Otherwise, copy it from the remote file.

<filekey> is the cache index key. A good choice is (datasetDN
where datasetDN is the distinguished name of the dataset, and
is the name of the file within the dataset.

For request manager transfers, <lcpath> is the logical collec
<userid> is the user string ID, <useReplica> is true iff the
search the replica catalog for the actual file to transfer.

Returns the path of a file in the cache.

Note: The function does not guarantee that the file is still
by the time it returns.

put(self, filekey, path)

```
cache[filekey] = path
```

Data and other attributes defined here:

indexpath = None

Functions

copyFile(fromURL, toURL, callback=None, lcpath=None, userid=None, useReplica=1)

Copy file <fromURL> to local file <toURL>. For FTP transfers, if c
display a progress dialog, otherwise just print progress messages.

For request manager transfers, <lcpath> is the logical collection
<userid> is the string user ID, <useReplica> is true iff the reque
search the replica catalog for the actual file to transfer.

lock(filename)

Acquire a file-based lock with the given name.
Usage: lock(filename)
If the function returns, the lock was acquired successfully.
Note: This function is UNIX-specific.
Note: It is important to delete the lock via unlock() if the proce
  is interrupted, otherwise subsequent locks will fail.

lockpath(filename)

Generate the pathname of a lock. Creates the directory containing
if necessary.
Usage: lockpath(filename)

unlock(filename)

```
Delete a file-based lock with the given name.
Usage:unlock(filename)
If the function returns, the lock was successfully deleted.
Note: This function is UNIX-specific.
```

useGlobusTransfer()

```
Specify that file transfers should use the Globus storage API (SC-
```

usePythonTransfer()

```
Specify that file transfers should use the Python libraries urllib
```

useRequestManagerTransfer()

useTTY()

```
Informational messages such as FTP status should be sent to the te
```

useWindow()

```
Specify that dialog windows should be used if possible. Do not cal
gui.setProgressParent instead. See useTTY.
```

Data

GlobusNotSupported = 'Globus interface not supported'
LockError = 'Lock error:'
MethodNotImplemented = 'Method not yet implemented'
RequestManagerNotSupported = 'Request manager interface not supported (module reqm not found)'
SchemeNotSupported = 'Scheme not supported: '
TimeOutError = 'Wait for read completion timed out:'